

REMARKS

Reconsideration of this application, based on this amendment and these following remarks, is respectfully requested.

Claims 2 through 17 remain in this case. No claims are amended.

The Abstract is amended as required by the Examiner. There are now fewer than 150 words in the Abstract, as amended, even counting the articles "a", "an" and "the". Applicant respectfully submits that the Abstract now meets the suggestions of the Rules of Practice, in their current form.¹

The title is amended, as requested by the Examiner.

Claims 2 through 4, 6, and 12 through 15 were again rejected under §103 as unpatentable over the Kiuchi et al. reference² in view of the Osovets reference³. Claims 5, 7, 8, 16, and 17 were finally rejected over the Kiuchi et al. and Osovets reference as applied against the above claims, and further in view of the George reference⁴. Claims 9 through 11 were finally rejected under §103 as unpatentable over the Kiuchi et al. reference in view of the Hennessy reference⁵.

The stated bases for the rejection of the claims in this case remain the same as previously urged by the Examiner, for example in the Office Action of September 21, 2004. Accordingly, much of the same argument will be restated in this paper, at the risk of redundancy. In the current Office Action, however, the Examiner responds to Applicant's previous argument; the

¹ 37 C.F.R. §1.72.

² U.S. Patent No. 5,579,493, issued November 26, 1996 to Kiuchi et al..

³ U.S. Patent No. 6,125,440, filed May 21, 1998, and issued September 26, 2000 to Osovets.

⁴ U.S. Patent No. 4,626,988, issued December 2, 1986 to George.

⁵ Hennessy et al., *Computer Organization and Design - The Hardware/Software Interface* (2d. ed., Morgan Kaufmann Publishers, Inc., 1998), pp. 542, 545, 549, 579.

following argument will specifically address the Examiner's response, in the context of Applicant's patentability position.⁶

Applicant again respectfully traverses the rejection of claim 2 and its dependent claims.

Claim 2 is directed to an instruction-programmable processor having, *inter alia*, loop cache control logic for controlling the branch cache register file to load an instruction code received at its data input from the program memory responsive to receiving a backward branch signal in combination with the fetch address not corresponding to one of the instruction codes stored in the branch cache register file. According to the invention of claim 2, therefore, the loop cache control logic enables the loading of instructions in the branch cache register file in response to backward branches, for example as a "loop front cache"⁷ or as a "loop tail cache"⁸ for which the instruction codes are not already stored in the loop cache (i.e., the "miss" condition of the fetch address not matching an instruction code already stored in the branch cache register file). Because of this construction and operation, the processor of claim 2 and its dependent claims advantageously minimizes the number of accesses to memory, in an automatic manner even for nested program loops, and minimizes the loading, into its cache loop, instructions that will only be executed once.⁹

As previously argued, claim 2 clearly and expressly requires that its loop cache control logic is for controlling the branch cache register file to load an instruction code received at its data input from the program memory responsive to receiving a backward branch signal in combination with the fetch address not corresponding to one of the instruction codes stored in the branch cache register file.

⁶ As Applicant is specifically reminded by the Examiner. Office Action of January 12, 2005, p. 27, § 30, citing 37 C.F.R. §1.111.

⁷ Specification of Application S.N. 09/713,731, page 14, line 20 *et seq.*; page 20, line 20 through page 21, line 28.

⁸ Specification, *supra*, page 22, line 15 *et seq.*; page 32, line 3 through page 33, line 2.

⁹ Specification, *supra*, page 5, lines 2 through 12; page 14, lines 20 through 26; page 21, lines 22 through 28; page 32, line 13 through page 33, line 8.

Applicant agrees with the Examiner that the Kiuchi et al. reference fails to disclose the loading and storing of instruction codes in a branch cache register file. As previously urged, however, Applicant submits that the Osovets reference fails to disclose the loading of the branch cache register file with an instruction code responsive to receiving a backward branch signal in combination with a fetch address not corresponding to one of the instruction codes stored in the branch cache register file, as required by claim 2.

In his response to Applicant's arguments, the Examiner asserted several dictionary definitions for the word "load", specifically "to place into internal storage", "to copy ... data ... to internal storage", and "to enter data ... into storage or working registers".¹⁰ These dictionary definitions were three definitions that were selected from a total of seventeen provided for the word "load". Based on these selected definitions, the Examiner circularly asserted that the word "load" means "to place data into storage or registers, which is then inherently retained for some time", and conversely, "to retain a value ... the data must be placed or loaded there".¹¹ And using this interpretation of "load", the Examiner then equates the retaining of repeated or looped instructions in a shift register as taught by the Osovets reference to the load operation of the branch cache register file recited in claim 2.

Applicant accepts, for purposes of this argument, definition 9B asserted by the Examiner for the word "load", namely:

(B) (software) To copy computer instructions or data from external storage to internal storage or from internal storage to registers. *Contrast: store. See also: fetch; move.*¹²

However, at the risk of repetition, the operative phrase of claim 2 using the word "load" is:

the loop cache control logic also for controlling the branch cache register file to load an instruction code received at its data input from the program memory responsive to receiving a backward branch signal in combination with the fetch

¹⁰ Office Action, *supra*, page 6, §8.c., citing IEEE 100: *The Authoritative Dictionary of IEEE Standards Terms* (7th ed., 2000), p. 629(7).

¹¹ *Id.*

¹² IEEE 100, *supra* (italics in original).

address not corresponding to one of the instruction codes stored in the branch cache register file¹³

It is self-evident from the claim that the act of controlling the branch cache register file to load (*i.e.* copy into itself, according to definition 9B) an instruction code received at its data input from the program memory, is to be carried out responsive to the received backward branch signal in combination with the fetch address not corresponding to one of the instruction codes stored in the branch cache register file.¹⁴

The Examiner errs, in his rejection of claim 2, by ignoring the "responsive to" limitation present in the claim. This error is evident from the Examiner's circular analysis, which, to paraphrase, is: because to "load" inherently results in "retaining" of the loaded information, and because to "retain" information necessitates that the information was at some time previously "loaded", then "retain" necessarily encompasses "load". But the fallacy of this analysis is shown by the Examiner's assertion that "Osovets has shown that the repeated or looped instructions are retained in the shift register . . . once a jump back or branch backwards instructions[sic] is decoded"¹⁵. Merely because the decision to "retain" previously loaded instructions is conditional, does not make the *loading* of those instructions responsive to the same condition, or to any condition. Rather, as evident from the Osovets reference itself and as previously argued by Applicant, the Osovets reference teaches the loading of *every* instruction code into its shift register, regardless of whether a backward branch has been taken, and the retaining (without loading new instruction codes) of those instruction codes upon detecting a backward branch.

In contrast, if one properly applies dictionary definition 9B, presented by the Examiner himself, to the limitation in claim 2 "to load ... responsive to receiving a backward branch signal in combination with the fetch address not corresponding to one of the instruction codes stored in the branch cache register file", it is apparent that it is the *act* of loading that is

¹³ Application S.N. 09/713,731, claim 2 (as amended).

¹⁴ See also specification, *supra*, page 20, line 20 through page 21, line 9; page 28, lines 12 through 23; page 30, lines 8 through 25.

¹⁵ Office Action, *supra*.

conditioned on the "responsive to" condition of the claim. This conditional loading is nowhere disclosed by the Osovets reference, nor by the Kiuchi et al. reference as previously argued. Indeed, as previously argued, this recited operation is the opposite of that taught by the Osovets reference, in which loading continues unconditionally until a backward branch instruction, at which point the loading stops.

The Examiner attempts to finesse this point by arguing that the Osovets reference does not teach the loading of instruction code unconditionally, but that the reference instead teaches that the instruction codes "in response to encountering them . . . upon encountering any instruction its code is loaded . . . [and u]pon encountering a backwards branch instruction then, the instruction code is loaded."¹⁶ The Examiner further asserts that "[t]he claim language does not limit the scope to say that only backwards branch instructions are loaded and all other instructions are not loaded".¹⁷

Perhaps claim 2 does not expressly state that "all other instructions are not loaded". But the claim *does* state, as urged above, the condition under which instruction codes *are* loaded in the claimed processor, namely "responsive to receiving a backward branch signal in combination with the fetch address not corresponding to one of the instruction codes stored in the branch cache register file". Words have no meaning,¹⁸ and indeed this express claim limitation would have no meaning, if a prior art teaching that "upon encountering any instruction its code is loaded" meets a claim limitation requiring that an instruction code is loaded responsive to receiving a specified signal in combination with a particular relationship of an instruction address to stored values. In other words, according to the Examiner's approach, claim 2 would have the same scope if its "responsive to" limitation were not present. Fortunately, under the patent law, claim limitations do have meaning.¹⁹

¹⁶ Office Action, *supra*, page 26, §28.

¹⁷ *Id.*

¹⁸ With apologies to Justice Scalia, *Roper v. Simmons*, 543 U.S. ___, slip opinion at 3, (2005) (Scalia, J., dissenting).

¹⁹ See *In re Wilson*, 424 F.2d 981, 180 USPQ 580 (CCPA 1974), MPEP §2143.03.

Accordingly, neither of the Kiuchi et al. and Osovets references teach the controlling of the branch cache register file by the loop cache control logic as required by claim 2. And none of the other references of record disclose this function. Therefore, Applicant respectfully submits that the combined teachings of the applied references fall short of the requirements of claims 2 through 11, and that the Examiner has failed to present a *prima facie* case that these claims are unpatentable. The rejection of claims 2, 3, 4, and 6 is therefore in error.

And as previously argued, Applicant further respectfully submits that there is no suggestion to modify these teachings in such a manner as to reach claim 2 and its dependent claims. The Kiuchi et al. reference requires the programmer to use a specific "repeat" instruction for its operation. While the Osovets reference does not require such an instruction, each fetched instruction is simply unconditionally loaded into also the shift register bank, and the backward branch then enables *access* of these stored instructions from the shift register bank. The George reference, asserted against claims 5, 7, and 8, and the Hennessey reference, asserted against claims 9 through 11, were not cited as provided any teachings in this regard, in fact they provide no such teachings. There is no disclosure by or suggestion from the prior art of record in this case to carry out the act of loading the instruction codes into the branch cache register file responsive to a backward branch, as required by claim 2 and its dependent claims.

The claimed processor of claim 2 provides the important advantages that it can to closely cache the instructions for nested program loops, and minimizes the caching of instructions that are executed only once. These advantages arise directly from the difference between the claim and the prior art, which negates suggestion from the prior art to modify its teachings in such a manner as to reach the claim, and indicates the value of the inventive processor.

For these reasons, Applicant respectfully submits that claim 2 and its dependent claims are patentably distinct over the applied references, taken individually or in any proper combination. The rejection of claim 2 and its dependent claims is therefore respectfully traversed.

Applicant further respectfully submits that dependent claims 6 through 8 are further patentably distinct relative to the applied references.

As previously argued, claim 6 further requires, relative to claim 2 upon which it depends, that the base address register is for loading the contents of the next candidate register as a base fetch address responsive to the loop cache control logic receiving a backward branch signal in combination with the fetch address corresponding to the contents of the next candidate register, and that the loop cache control logic is for controlling the branch cache register file to load an instruction code received at its data input from the program memory responsive to receiving a backward branch signal in combination with the fetch address corresponding to the contents of the next candidate register.

For the reasons discussed above relative to claim 2, Applicants submit that the references of record nowhere disclose the loading of instruction codes in the branch cache register file in response to a backward branch in combination with the fetch address not corresponding to one of the instruction codes stored in the branch cache register file. Absent any disclosure of this limitation, there is necessarily no disclosure by those references of the additional limitations of claim 6, relative to its next candidate register and its additionally recited functions of the base address register and the loop cache control logic. And absent suggestion in the prior art to modify those teachings to provide the responsive condition under which loading of instruction codes responsive to a backward branch is carried out, as discussed above, the prior art especially fails to further modify those teachings to provide the "loop tail cache" functionality that results from the construction of claims 6 through 8.

For these reasons, Applicant respectfully submits that claims 6 through 8 are further patentably distinct over the prior art of record in this case.

Claims 9 through 11 were specifically rejected as unpatentable over the Kiuchi et al. reference in view of the Hennessey reference.²⁰

²⁰ Office Action, *supra*, pages 23 through 25.

Applicant respectfully traverses the rejection of claims 9 through 11. As mentioned above, the Examiner looked to the Osovets reference (albeit erroneously) to find the recited loading of instruction codes into a shift register in order to meet the limitations of claim 2. However, the Osovets reference is not included in the basis of rejection of claims 9 through 11, despite claims 9 through 11 depending on claim 2. And the rejection nowhere asserts that the recited functionality of the loop cache control logic discussed above relative to claim 2 is taught or suggested by the Hennessy reference. Instead, the Hennessy reference is asserted as teaching a level one tag memory. Accordingly, the rejection of claims 9 through 11, on the grounds stated in the Office Action, is clearly not based on a properly established *prima facie* case of obviousness of these claims, and the rejection is therefore in error.

And for the same reasons as discussed above relative to claim 2, upon which these claims depend, Applicant respectfully submits that the combined teachings of the Kiuchi et al. and Hennessy references, as well as the Osovets reference, fall short of the requirements of claims 9 through 11. Applicants therefore respectfully submit that claims 9 through 11 are patentably distinct over the prior art of record in this case.

Applicant also respectfully traverses the rejections of claim 12 and its dependent claims.

As argued previously and above relative to claim 2, Applicant submits that the Kiuchi et al. and Osovets references both fail to disclose the loading of a branch cache register file with an instruction code responsive to receiving a fetch address following a backward branch operation and within a storage capacity of the branch cache register file, as required by claim 12.

The Kiuchi et al. reference was not asserted as disclosing this step. And, as mentioned above, Applicant submits that the Osovets reference fails to disclose this step. The Osovets reference instead teaches loading every instruction code into its shift register, regardless of whether a backward branch has been taken; it is the retaining of instruction codes that have already been loaded into the shift register and the inhibiting of the loading of new instruction codes that are instead performed in response to detecting a backward branch.

The rejection of claim 12 is based on the same erroneous approach taken by the Examiner in his rejection of claim 2, specifically by ignoring the "responsive to" limitation present in the claim. Specifically, claim 12 requires the step of loading an instruction code, into a next indexed location of a branch cache register file, responsive to receiving a fetch address following a backward branch operation and within a storage capacity of the branch cache register file. The Examiner again asserts that "the repeated or looped instructions are stored in the shift register (branch cache register file) once a jump back or branch backwards instruction is decoded".²¹ However, the Examiner nowhere asserts that the recited step of loading the branch cache register file is performed responsive to the events recited in claim 12, or responsive to any condition or event. Rather, as discussed above, the Examiner apparently follows the rationale applied against claim 2, in that the Osovets reference does not teach the loading of instruction code unconditionally, but that the reference instead teaches that the instruction codes "in response to encountering them . . . upon encountering any instruction its code is loaded . . . [and u]pon encountering a backwards branch instruction then, the instruction code is loaded."²² But to apply this teaching against claim 12 requires one to ignore the "responsive to" limitation of the loading step of claim 12, which is of course improper under the law.²³

Accordingly, neither of the Kiuchi et al. and Osovets references, nor any of the other prior art of record in this case, teaches the loading of a next indexed location of a branch cache register file with an instruction code responsive to receiving a fetch address following the backward branch operation and within a storage capacity of the branch cache register file, as required by claim 12. Accordingly, Applicant respectfully submits that, upon entry of this amendment, the combined teachings of the prior art of record in this case fall short of the requirements of claims 12 through 17.

Applicant again respectfully submits that there is no suggestion to modify these teachings in such a manner as to reach claim 12 and its dependent claims. The Kiuchi et al.

²¹ Office Action, *supra*, p. 11, §12 (b)(iii).

²² Office Action, *supra*, page 26, §28.

²³ See *Wilson, supra*; MPEP §2143.03.

reference lacks such suggestion, because it requires the use of a specific "repeat" instruction to enable its functionality, and the Osovets reference lacks such suggestion because of its unconditional loading of instruction codes into the shift register bank, with backward branches then enabling *access* of these stored instructions from the shift register bank. There is no suggestion from these references, nor from the other prior art, to load the instructions in the branch cache register file responsive to any backward branch, as required by claim 12 and its dependent claims.

The advantages provided by the processor of claim 12, especially in minimizing caching of instructions that are executed only once, are due directly to the difference between the claim and the prior art, and therefore support the patentability of these claims. Accordingly, Applicant respectfully submits that there is no suggestion in the prior art to modify the teachings of these references in such a manner as to reach the claim.

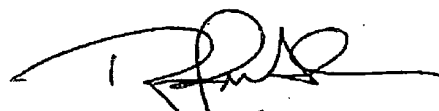
For these reasons, Applicant respectfully submits that claim 12 and its dependent claims are patentably distinct over the applied references, taken individually or in any proper combination. Applicant accordingly traverses the rejection of claim 12 and its dependent claims 13 through 17.

Applicant respectfully submits that claim 15 and its dependent claims 16 and 17 are further patentably distinct over the prior art of record in this case. As discussed above relative to claim 12, upon which claim 15 depends, the prior art fails to disclose the loading of instruction codes in a branch cache register file responsive to receiving a fetch address following the backward branch operation to within a storage capacity of the branch cache register file. Absent this teaching, there is necessarily no disclosure in the prior art of the additional limitations of claim 15, especially the comparing of a fetched address to the contents of a candidate register and the additionally recited functions of the loading the candidate register. Furthermore, considering that the prior art lacks any suggestion to load instruction codes responsive to a backward branch, as discussed above, there is especially no suggestion from the prior art to modify its combined teachings to provide the "loop tail cache" functionality that results from the construction of claim 15 and its dependent claims.

Applicant therefore respectfully submits that claims 15 through 17 are further patentably distinct over the prior art of record in this case.

For these reasons, Applicant respectfully submits that all claims in this case are in condition for allowance. Reconsideration of this application is therefore respectfully requested.

Respectfully submitted,



Rodney M. Anderson

Registry No. 31,939

Attorney for Applicant

Anderson, Levine & Lintel, L.L.P.

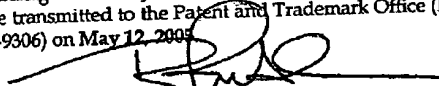
14785 Preston Road, Suite 650

Dallas, Texas 75254

(972) 664-9554

CERTIFICATE OF FACSIMILE TRANSMISSION
37 C.F.R. 1.8

The undersigned hereby certifies that this correspondence is being facsimile transmitted to the Patent and Trademark Office (Fax Number 703-872-9306) on May 12, 2005.



Rodney M. Anderson
Registry No. 31,939